Android IPC 简介

IPC 是 Inter-Process Communication 的 缩写，含义是进程间通信或者跨进程通信，是指两个进程之间进行数据交换的过程。说起进程间通信，我们首先要了解什么是进程，什么是线程，进程和线程是两个完全不同的概念。按照操作系统中的描述，线程是CPU调度的最小单元，同时线程是一种和有线的系统资源。

而进程一般指一个执行单元，在PC和移动设备上指一个程序或者一个应用。一个进程可以包含多个线程，因此进程和线程是包含和被包含的关系。最简单的情况下，一个进程可以只有一个线程，即主线程，在Android中主线程也叫UI线程，在UI线程中才能操作界面元素。

很多时候，一个进程中需要执行大量耗时的任务，如果这些任务放在主线程中去执行就会造成界面无法响应，严重影响用户的体验。这种情况在PC系统和移动系统中都存在，再Android中有一个特殊的名字叫做ANR（Application Not Response）即应用无响应。解决这个问题就需要用到线程，把一些耗时的操作放在线程中即可。

IPC不是Android中所独有的，热河一个操作系统都需要有相对应的IPC机制比如Windows上可以通过剪贴板、管道和邮槽等来进行进程间通信；Linux上可以通过命名管道、共享内存、信号量等来进行进程间通信。可以看到不同的操作系统平台有着不同的进程间通信的方式，对于Android来再说，它是一种基于Linux内核的移动操作系统，它的进程间通信并不能完全继承自Linux，相反，它有自己的进程间通信的方式。在Android中最具有特色的进程间通信方式就是Binder了，通过Binder可以轻松地实现进程间通信。除了Binder，Android还支持Socket，通过Socket也可以实现任意两个终端之间的通信。一个设备上的两个进程通过Socket通信自然也是可以的。

说道IPC的使用场景就必须要提到多进程，只有面对多进程这种场景，才需要考虑进程间通信。这个是很好理解的，如果只有一个进程在运行，又何谈多进程呢？多进程的情况分两种。第一种情况是一个应用因为某些原因自身需要采用多进程模式来实现，至于原因，可能有很多，比如有些模块由于特殊的原因需要运行在单独的进程中，又或者是为了加大一个应用可使用的内存所以需要多进程来获取多份内存空间。Android对单个应用所使用的内存大小做了限制，早期的一些版本可能是16MB，不同的设备有不同的大小。另一种情况是当前的额应用需要向其他应用获取数据，由于是两个应用，所以必须采用跨进程的方式来获取所需的数据，甚至我们通过系统提供的ContentProvider 去查询数据的时候，也是一种进程间通信，只不过通信细节被系统内部屏蔽了，我们无法感知而已。

总之，不管由于何种原因，我们采用了多进程的设计方法，那么应用中就必须妥善的处理进程间通信的问题。

所有运行在不同进程中的四大组件，只要它们之间需要通过内存来共享数据，都会共享失败，这也是多进程所带来的主要影响。正常情况下，四大组件中间不可能通过中间层来共享数据，那么通过简单的指定进程名来开启多线程都会无法正确运行。当然，特殊情况下，某些组件之间不需要共享数据，这个时候可以直接指定 Android:process属性来开启多线程，但这种场景是不常见的，几乎所有的情况都是需要共享数据。

如果要用一句话来形容多进程，我我只能这样说：“当应用开启了多进程以后，各种奇怪的问题都出现了”。为什么这样说呢？这是有原因的。大部分人认为开启多进程是很简答的事情，只需要给四大组件制定 android: process属性即可。比如说在实际的产品开发中，可能会有多进程的需求，需要把某些组件放在单独的进程中进行，然后迅速的给那些组件指定了 android:proccess属性，然后编译运行，发现“正常的运行起来了”。真的是这样吗？

一般来说，使用多进程会造成如下几方面的问题：

（1）静态成员和单例模式完全失效；

（2）线程同步机制完全失效；

（3）SharePreferences的可靠性下降；

（4）Application会多次创建。

所以，大家明白了这就是多进程所带来的问题，多进程绝非仅仅是指定一个android:pcocess属性那么简单。

Introduction to Android IPC

IPC is the abbreviation of Inter-Process Communication, meaning inter-process communication or cross-process communication, refers to the process of data exchange between the two processes. Speaking of interprocess communication, we must first understand what is the process, what is the thread, the process and the thread is two completely different concepts. According to the description in the operating system, the thread is the smallest unit of CPU scheduling, while the thread is a wired and system resources.

The process generally refers to an execution unit, in the PC and mobile devices refer to a program or an application. A process can contain multiple threads, so processes and threads are inclusive and included. In the simplest case, a process can have only one thread, the main thread, the main thread in Android, also called UI thread, in the UI thread to operate the interface elements.

In many cases, a process requires a lot of time-consuming tasks, if these tasks on the main thread to the implementation will cause the interface can not respond, seriously affect the user experience. This situation exists in the PC system and mobile systems, and then there is a special name in Android called ANR (Application Not Response) that application is no response. To solve this problem need to use the thread, some time-consuming operation can be placed in the thread.

IPC is not unique in Android, Jehol an operating system need to have a corresponding IPC mechanism such as Windows can be through the clipboard, pipeline and postal mail for inter-process communication; Linux can be named through the pipeline, shared memory , Semaphores, etc. to carry out interprocess communication. You can see a different operating system platform has a different way of communication between the way, for Android to say that it is a Linux kernel-based mobile operating system, its inter-process communication can not be completely inherited from Linux, on the contrary, it has Their way of communication between processes. The most unique inter-process communication in Android is Binder, and it is easy to implement inter-process communication with Binder. In addition to Binder, Android also supports Socket, through the Socket can also achieve any two terminals between the communication. A device on the two processes through the Socket communication is naturally possible.

Said the use of IPC scene must be mentioned in the multi-process, only the face of this process of multi-process, only need to consider inter-process communication. This is a good understanding, if only one process is running, and how to talk about the process? There are two cases of multiple processes. The first case is an application for some reason itself needs to use multi-process mode to achieve, as for the reasons, there may be many, such as some modules for special reasons need to run in a separate process, or in order to increase an application Can use the memory so you need multiple processes to get multiple memory space. Android has limited the size of memory used for a single application. Some of the earlier versions might be 16MB, and different devices have different sizes. Another situation is the current amount of application needs to obtain data from other applications, because it is two applications, it must be cross-process approach to obtain the required data, even when we provide the ContentProvider to query the data, A process of communication, but the details of the communication system is shielded, we can not perceive it.

In short, for whatever reason, we use a multi-process design method, then the application must be properly handled between the process of communication problems.

All four components running in different processes, as long as they need to share data through memory, will share the failure, which is the main impact of multi-process. Under normal circumstances, the four components can not be shared through the middle of the middle of the data, then simply specify the process name to open the multi-threaded will not run correctly. Of course, special circumstances, some components do not need to share data between, this time you can directly specify the Android: process attribute to open multi-threaded, but this scene is not common, almost all of the situation is the need to share data.

If you want to use a word to describe the process, I can only say: "When the application has opened a multi-process, all kinds of strange problems have emerged." Why do you say this way? There is a reason for this. Most people think that the process is very simple to open a few things, only need to develop four components android: process attributes can be. For example, in the actual product development, there may be more process needs, the need to put some components in a separate process, And then quickly assigned to those components android: proccess attribute.Then compile and run, Normal operation up ". Is it really?

In general, the use of multi-process will cause the following aspects:

(1) static member and singleton mode are completely disabled;

(2) thread synchronization mechanism completely failed;

(3) SharePreferences the reliability of the decline;

(4) Application will be created many times.

So we understand that this is the process of the problems brought about by the multi-process is not just to specify an android: pcocess attribute so simple.